J.W. PO (714) -84

MAIL ROOM MAY 26 1994 PAT. & TRADEMARK OFF.

NAKI-AN73
MASATO SUZUKI et al.

本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: １９９３年　５月２７日

出願番号
Application Number: 平成　５年特許願第１２６２１２号

出願人
Applicant (s): 松下電器産業株式会社

１９９４年　３月１１日

特許庁長官
Commissioner,
Patent Office 麻生渡

出証番号　出証特平０６－３００８６４１

【書類名】　特許願

【整理番号】　2030452090

【提出日】　平成 5年 5月27日

【あて先】　特許庁長官　麻　生　渡　殿

【国際特許分類】　G06F　9/44

【発明の名称】　プログラム変換装置およびプロセッサ

【請求項の数】　5

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　鈴木　正人

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　神山　祐史

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　宮地　信哉

【特許出願人】

【識別番号】　000005821

【氏名又は名称】　松下電器産業株式会社

【代表者】　森下　洋一

【代理人】

【識別番号】　100090446

【弁理士】

【氏名又は名称】　中島　司朗

【手数料の表示】

【納付方法】　予納

【予納台帳番号】　014823

【納付金額】　14,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9003742

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　　プログラム変換装置およびプロセッサ

【特許請求の範囲】

【請求項１】　主たるデータの型における有効なビット幅Ｍよりも長いビット幅Ｎをもつアドレスを扱うプロセッサを対象とし、高級言語プログラムに基づいて機械語命令を生成するプログラム変換装置であって、

主たるデータの型における有効なビット幅およびアドレスのビット幅を保持するパラメータ保持手段と、

生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成し、アドレスを表す変数である場合、パラメータ保持手段が保持するアドレスのビット幅を有効とする命令を生成する生成手段と

を備えたことを特徴とするプログラム変換装置。

【請求項２】　前記データのビット幅Ｍは１６であり、

前記アドレスのビット幅Ｎは１７以上３１以下である

ことを特徴とする請求項１記載のプログラム変換装置。

【請求項３】　前記生成手段は、

生成すべき機械語命令がメモリをアクセスする命令である場合に、アクセス対象となる変数がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅をアクセス幅とすべきことを指示するメモリ管理手段と、

生成すべき機械語命令がレジスタを使用する命令である場合に、そのレジスタにリード又はライトされる変数がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅を有効とすべきことを指示するレジスタ管理手段と、

生成すべき機械語命令が即値を使用する命令である場合にその即値がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅の即値を用いるべきことを指示する即値管理手段と

メモリ管理手段、レジスタ管理手段及び即値管理手段からの指示に従って、機

械語命令を生成するコード生成手段と
からなることを特徴とする請求項１又は２記載のプログラム変換装置。

【請求項４】　請求項１ないし３記載のいずれかのプログラム変換装置により生成されたプログラムを実行するプロセッサであって、

Ｎビット長のプログラムカウンタと、

Ｎビット長の演算、及びデータのビット幅であるＭビット長の演算を実行する実行手段と
を備えたことを特徴とするプロセッサ。

【請求項５】　前記Ｎは２４である
ことを特徴とする請求項３記載のプロセッサ。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】

本発明は高級言語プログラムを機械語プログラムに変換するプログラム変換装置、及び、その機械語プログラムを実行するプロセッサに関する。

【０００２】

【従来の技術】

近年の電子技術の発展により、マイクロプロセッサ等の情報処理装置が普及し、あらゆる分野で用いられている。。

従来のプロセッサとしては、例えば、森下巖著、「マイクロプロセッサのハードウェア」（１９８４年１１月９日、岩波書店）に示されている、セグメントアドレス方式の１６ビットプロセッサがある。

【０００３】

このセグメントアドレス方式の１６ビットプロセッサは、１６ビットよりも上位のビットを含む上位アドレスを格納するセグメントレジスタを備えることにより、データのビット幅が１６ビットでありながら６４キロバイトを超えるアドレス空間を扱うことができる。すなわち、６４キロバイトを越える全アドレス空間をセグメントと呼ぶ６４キロバイト単位の部分に分けるとともに、各セグメントに番号を付ける方式である。アドレスは、セグメントレジスタに収められたセグ

メント番号と、データのビット幅と同じ１６ビットで表されるセグメントの先頭からの距離（オフセット）とで管理される。これにより、セグメントアドレス方式の１６ビットプロセッサでは、データのビット幅が１６ビットで６４キロバイトを超えるアドレス空間を扱うことができる。

【０００４】

また、従来のプロセッサとして、同書に示されている３２ビットプロセッサがある。この３２ビットプロセッサは、３２ビット幅のアドレスを３２ビット幅のデータと同一に管理して、４ギガバイトのアドレス空間を扱うことができる。

このようなプロセッサ上で実行される機械語プログラムは、コンパイラ等のプログラム変換装置によって生成される。

【０００５】

上記１６ビットプロセッサに対するコンパイラは、アドレス管理の方式に関して、ラージモデル方式を呼ばれるものと、ニアファーモデル方式と呼ばれるものがある。

ラージモデル方式を採用するコンパイラは、ポインタ変数を常にセグメントレジスタと１６ビットオフセットとを対にして設定する。このため、このコンパイラによるオブジェクトコードを実行するプロセッサは、アドレス計算の度にセグメントレジスタの内容を計算して更新する操作が発生し、セグメントアドレス方式を採らない１６ビットプロセッサに比べて著しく性能が劣化するという短所を有している。

【０００６】

また、ニアファーモデル方式とを採用するコンパイラは、ラージモデル方式における上記の短所を次のようにして解決している。この方式は、同一セグメント内をアクセスするためのニアポインタ変数と、セグメント境界をまたいでアクセスするためのファーポインタ変数という２種類のポインタ変数を指定できるようになっている。コンパイラは、ニアポインタ変数には１６ビットオフセットのみを設定し、ファーポインタ変数にはセグメントレジスタと１６ビットオフセットとを対にして設定する。ニアポインタ変数とファーポインタ変数のいずれを用いるかは、プログラマが選択する。

【０００７】

このニアファーモデル方式は、ニアポインタ変数を用いる場面ではラージモデル方式に比べて性能が向上する反面、プログラマがプログラム作成時にセグメント境界を認識して２種類のポインタ変数を使い分ける必要があり、プログラム開発効率が著しく低下するという新たな問題点を有している。

他方、３２ビットプロセッサに対するコンパイラは、上記の問題点を有していない。即ち、コンパイラが３２ビットのデータ変数と同様にポインタ変数にも３２ビットのアドレスを設定するので、プログラマがセグメントの境界を認識する必要がないので、プログラムの開発効率を低下させることもない。また、コンパイルされたプロクラムを実行するプロセッサは、アドレス計算による性能劣化を発生させない。そして、４ギガバイトという広大なアドレス空間を扱うことができる。

【０００８】

【発明が解決しようとする課題】

しかしながら従来技術のプロセッサおよびプログラム変換装置によれば、組み込み用途のマイクロコンピュータでの応用（アプリケーション）おいては、３２ビットのデータ幅を必要とせず、かつ、４ギガバイトのアドレス空間を必要としないものがほとんどである。多くのアプリケーションでは、データ幅については１６ビットあれば十分であるが、アドレス空間については６４キロバイトでは不足する。このような現状に鑑みると、データのビット幅が１６ビット必要で、かつ、アドレス空間が６４キロバイト以上を必要とするが４ギガバイトもの大きさを必要としない応用では、最適なマイクロコンピュータおよびコンパイラがなかったという問題点があった。

【０００９】

より具体的に説明すると、３２ビットプロセッサは、３２ビットのデータおよびアドレスを処理するためのハードウェアを有することから、過分なハードウェアとなるのでコストおよび消費電力が増大するという問題点を有している。１６ビットプロセッサは、アドレス空間の管理が従来技術で述べたようにアドレス計算の点で性能劣化を避けられない。また、プログラマがセグメントの境界を認識

する必要があることから、プログラムの開発効率が落ちる。

【００１０】

また、３２ビットプロセッサ用コンパイラは、１６ビットまたは３２ビットを基本語長とする機械語命令を使い分け、プログラム中では３２ビットのアドレス指定が必要なために、プログラムのコードサイズが増大する。また、３２ビットプロセッサとメモリとを接続するデータバスのビット幅が１６ビットの場合は３２ビットに比べて性能が著しく劣化するという問題点も有している。

【００１１】

本発明は上記問題点に鑑み、データのビット幅が１６ビットで１６メガバイト程度のアドレス空間を必要とする応用において、好適したプログラム変換装置およびプロセッサを提供することを目的とする。

【００１２】

【課題を解決するための手段】

上記の課題を解決するため本発明のプログラム変換装置は、

主たるデータの型における有効なビット幅よりも長いビット幅をもつアドレスを扱うプロセッサを対象とし、高級言語プログラムに基づいて機械語命令を生成するプログラム変換装置であって、

主たるデータの型における有効なビット幅およびアドレスのビット幅を保持するパラメータ保持手段と、

生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成し、アドレスを表す変数である場合、パラメータ保持手段が保持するアドレスのビット幅を有効とする命令を生成する生成手段と

を備えている。

【００１３】

前記データのビット幅Mは１６であり、前記アドレスのビット幅Ｎは１７以上３１以下であってもよい。

また、前記生成手段は、

生成すべき機械語命令がメモリをアクセスする命令である場合に、アクセス対

象となる変数がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅をアクセス幅とすべきことを指示するメモリ管理手段と、

生成すべき機械語命令がレジスタを使用する命令である場合に、そのレジスタにリード又はライトされる変数がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅を有効とすべきことを指示するレジスタ管理手段と、

生成すべき機械語命令が即値を使用する命令である場合にその即値がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅の即値を用いるべきことを指示する即値管理手段と

メモリ管理手段、レジスタ管理手段、即値管理手段からの指示に従って、機械後命令を生成するコード生成手段と

からなっていてもよい。

【００１４】

また、本発明のプロセッサは、

上記プログラム変換装置により生成されたプログラムを実行するプロセッサであって、

Ｎビット長のプログラムカウンタと、

Ｎビット長の演算、及びデータのビット幅であるＭビット長の演算を実行する実行手段と

を備えている。

【００１５】

ここで、前記Ｎは２４であり、Ｍは１６であってもよい。

【００１６】

【作用】

上記の手段により本発明のプログラム変換装置において、生成手段は、生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成する。アドレスを表す変数である場合、パラメータ保持手段が保持するアドレスのビット幅を有効

とする命令を生成する。

【００１７】

また、前記生成手段において、メモリ管理手段は、生成すべき機械語命令がメモリをアクセスする命令である場合に、アクセス対象となる変数がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅をアクセス幅とすべきことをコード生成手段に指示する。レジスタ管理手段は、生成すべき機械語命令がレジスタを使用する命令である場合に、そのレジスタにリード又はライトされる変数がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅を有効とすべきことをコード生成手段に指示する。即値管理手段は、生成すべき機械語命令が即値を使用する命令である場合にその即値がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅の即値を用いるべきことをコード生成手段に指示する。

【００１８】

また、本発明のプロセッサは、Ｎ（２４）ビット長のプログラムカウンタにより２のＮ乗（１６メガバイト）のアドレス空間を利用でき、実行手段によりＮ（２４）ビット長の演算、及びデータのビット幅であるＭ（１６）ビット長の演算を実行し得る。

【００１９】

【実施例】

図１は本発明の実施例におけるマイクロコンピュータおよびプログラム変換装置（以下、コンパイラと略す。マイクロコンピュータとコンパイラを含めて情報処理装置と呼ぶ）のブロック図を示す。図１において、本情報処理装置は、Ｃ言語プログラムを格納している記憶装置１と、Ｃ言語プログラムを機械語プログラムに翻訳するコンパイラ２と、オブジェクトコードを実行するハードウェア３とからなる。

【００２０】

コンパイラ２は、以下のものからなる。

５は構文解析部で、Ｃ言語プログラムの構文を解析し、プログラム文が中間形

式に変換された中間形式文及び中間コードからなる中間ファイルを生成する。

７はコード生成部で、中間ファイルに基づいてオブジェクトコードを生成する。

【００２１】

８はパラメータ設定部で、整数型のデータ変数のビット幅とポインタ変数のビット幅とを保持する。本実施例では、予め利用者により整数型のデータ変数のビット幅が１６ビットに、ポインタ変数のビット幅が２４ビットに設定されているものとする。

９はメモリ管理部で、コード生成部７が生成すべきロード・ストア命令について、その命令の対象となる変数の型に応じて、メモリを読み書きするビット幅を指示する。

【００２２】

１０はレジスタ管理部で、コード生成部７が生成すべきレジスタを使用する命令について、レジスタにリード／ライトされる変数の型に応じて、有効なビット幅を指示する。

１１は即値管理部で、コード生成部７が生成すべき即値を用いる命令について、即値である変数の型に応じて、命令コード中の即値のビット幅を指示する。

【００２３】

また、ハードウェア３は、以下のものからなる。

１３はメモリで、オブジェクトコードおよびプログラムで用いられるデータを格納する。

１４はアドレスバスで、２４ビットのビット幅を持つ。

１５はデータバスで、１６ビットのビット幅を持つ。

【００２４】

１６はバス制御回路で、アドレスバス１４とデータバス１５とを介してメモリ１３に接続され、メモリ１３に格納されたオブジェクトコードを命令として逐一読み出し、またメモリ１３に対してデータを読み書きする。

１７は命令制御回路で、プログラムカウンタ１９を有し、バス制御回路１６へ命令アドレスを指示し、バス制御回路１６から命令を受け取り解読する。

【００２５】

プログラムカウンタ１９は、２４ビットの命令アドレスを出力する。

１８は演算実行回路で、レジスタファイル２０と演算器２１からなる。

レジスタファイル２０は、２４ビットの複数のレジスタを有する。

演算器２１は、レジスタファイル２０に保持された値を用いて２４ビット幅の算術論理演算等を実行する。

【００２６】

図２は、コード生成部７の処理フローを示す。

ステップ２０は、中間ファイルからデータ変数を抽出して、それぞれの変数に関する情報を載せた変数テーブル（シンボルテーブル）を作成する。シンボルテーブルの一例を図７に示す。同図において、シンボル欄はデータ変数のシンボルを、型欄はデータ変数が整数型であるかポインタ型であるかを、バイト数欄はそのデータ変数のバイト数を、先頭アドレス欄はメモリに割り付けられている場合の先頭アドレスを、レジスタ欄はレジスタが割り当てれている場合のレジスタ名を示す。

【００２７】

ステップ２１は、中間ファイルからコード生成が未処理の中間命令があるか否かを判定する。あればステップ２２に進み、なければコード生成が終了する。

ステップ２２は、中間ファイルから次に処理すべき順の中間命令を１つ読み出す。

ステップ２３は、１つの中間命令を実現する１つ又は複数の機械語命令を選定する。

【００２８】

ステップ２４は、選定された機械語命令から次に処理すべき順の機械語命令を１つ指定する。

ステップ２５は、指定された１つの機械語命令（以下、個別命令と呼ぶ）の命令コードを生成する処理（以下、個別処理と呼ぶ）を行う。

ステップ２６は、ステップ２３で選定された機械語命令のうち次に処理すべき機械語命令があるか否かを判定する。あればステップ２４に戻り、なければステ

ップ２１に戻る。

【００２９】

図３～図６は、上記ステップ２５の個別処理のフローを示す。

第１に、ステップ３０は、個別命令がロード／ストア命令の種類に属するか否かを判定し、そうである場合はメモリ管理部９に通知し、そうでない場合ステップ３１に進む。この通知を受けたメモリ管理部９の処理フローを図４に示す。

図４において、ステップ４０は、その個別命令（ロード／ストア命令）においてメモリとの間でロード／ストアすべき変数について、シンボルテーブルを参照し、その変数の型を調べる。

【００３０】

ステップ４１は、その変数の型が整数型データ変数であるか、ポインタ型データ変数であるかを判定し、前者である場合ステップ４２に、後者である場合ステップ４３に進む。

ステップ４２は、アクセスするデータ幅が２バイトのロード／ストア命令を生成すべきことをコード生成部７に指示し、ステップ３１に進む。

【００３１】

ステップ４３は、アクセスするデータ幅が３バイトのロード／ストア命令を生成すべきことをコード生成部７に指示し、ステップ３１に進む。

第２に、ステップ３１は、個別命令がレジスタを使用するか否かを判定し、そうである場合はレジスタ管理部１０に通知し、そうでない場合ステップ３２に進む。この通知を受けた場合のレジスタ管理部１０の処理フローを図５に示す。

図５において、ステップ５０は、その個別命令が使用するレジスタに格納される変数について、シンボルテーブルを参照し、その変数の型を調べる。

【００３２】

ステップ５１は、その変数の型が整数型データ変数であるか、ポインタ型データ変数であるかを判定し、前者である場合ステップ５２に、後者である場合ステップ５３に進む。

ステップ５２は、使用するレジスタの下位１６ビットを有効とする命令を生成すべきことをコード生成部７に指示し、ステップ３２に進む。

【００３３】

ステップ５３は、使用するレジスタの全２４ビットを有効とする命令を生成すべきことをコード生成部７に指示し、ステップ３２に進む。

第３に、ステップ３２は、個別命令が即値データを使用するかか否かを判定し、そうである場合は即値管理部１１に通知し、そうでない場合ステップ３３に進む。この通知を受けた場合の即値管理部１１の処理フローを図６に示す。

【００３４】

図６において、ステップ６０は、当該個別命令において使用する即値データとなる変数について、シンボルテーブルを参照し、その変数の型を調べる。

ステップ６１は、その変数の型が整数型データ変数であるか、ポインタ型データ変数であるかを判定し、前者である場合ステップ６２に、後者である場合ステップ６３に進む。

【００３５】

ステップ６２は、２バイトの即値を持つ命令を生成すべきことをコード生成部７に指示し、ステップ３３に進む。

ステップ６３は、３バイトの即値を持つ命令を生成すべきことをコード生成部７に指示し、ステップ３３に進む。

第４に、ステップ３３は、メモリ管理部９、レジスタ管理部１０、即値管理部１１からの指示があれば、それにしたがって個別命令の命令コードを生成する。

【００３６】

以上のように構成された本実施例のマイクロコンピュータおよびプログラム変換装置について、以下その動作を説明する。

説明を簡潔にするために、記憶装置１に記憶されているＣ言語プログラムが次に示す場合を例にあげる。

```
main ( )
{
        int *a, b, c ;
        c = *a + b + 1 ;
}
```

構文解析部５は、記憶装置１からＣ言語プログラムを取り出し、構文を文法に照らして解析して中間形式の言語で記述された中間ファイルを生成する。この中間ファイルのイメージを以下に示す。ただし、ここでは便宜上、中間形式の記述を意味がわかるように書き直してある。

【００３７】

中間形式文１：　(int *a, b, c)

中間命令１：　t1:=*a

中間命令２：　t2:=t1+b

中間命令３：　t3:=t2+1

中間命令４：　c:=t3

中間形式文１は、宣言文int *a, b, c に対応し、中間命令１～４は、演算式c = *a + b + 1 に対応する。

【００３８】

これらの中間形式文、中間命令は、それぞれ以下のようにしてオブジェクトコードに変換される。

この中間ファイルが入力されると、コード生成部７は、中間ファイル中のデータ変数（宣言のないものも含めて）を抽出し、それぞれの変数の型を調査し、必要があればメモリに割り当て、図７に示したシンボルテーブルを作成する（図２のステップ２０）。

【００３９】

上の例では、中間形式文１から明示に宣言された変数*a,b,cが抽出される。ポインタ変数として宣言された変数*aは、パラメータ設定部８においてポインタ変数のビット幅が２４ビットに設定されているので、メモリ上に２４ビット（３バイト）の領域が確保され割り付けられる。また、整数型データ変数として宣言された２つの変数b,cは、パラメータ設定部８において整数型データ変数のビット幅が１６ビットに設定されているので、それぞれメモリ上に１６ビット（２バイト）の領域が確保され割り付けられる。ここでは、変数*a,b,cは、それぞれメモリの1000番地から３バイト、1004番地から２バイト、1006番地から２バイトの領域に割り当てられているものとする。ただし、1003番地の１バイトは空領域であ

る。

【００４０】

さらに、中間命令１～４から一時変数t1,t2,t3が抽出される。一時変数t1,t2,t3は、演算対象となる変数に合わせて、それぞれ整数型データ変数として扱われる。

これらの変数に関する情報がシンボルテーブルに書き込まれる。このシンボルテーブルの内容は、以後レジスタ割当等の変更があれば、その都度ダイナミックにその内容が書き換えられる。既に示した図７は、この時点におけるシンボルテーブルの内容である。この時点では、レジスタ欄及び一時変数の先頭アドレス欄は、割当がないので空欄のままである。

【００４１】

この後、コード生成部７は、中間命令のそれぞれについて、以下のようにして対応する機械語命令を生成する。

<< 中間命令１ : t1:=*a >>

コード生成部７は、中間ファイル中に処理すべき（未処理の）中間命令が残っているか否かを判定する（図２のステップ２１）。この時点では、中間命令１～４が残っているので、ステップ２２に進む。

【００４２】

次に、コード生成部７は、未処理の中間命令のうち先頭の中間命令を１つ読み出し（ステップ２２）、その中間命令を実現する１つ又は複数の機械語命令を選定する（ステップ２３）。

ステップ２３における中間命令１を実現する１つ又は複数の命令の選定を具体的に説明する。

【００４３】

中間命令１は「①ポインタ変数*aが割り当てられている1000番地から、②３バイトを読み出して、③その内容をアドレスとして２バイト読み出して一時変数t1に格納する。」ことを内容とする。コード生成部７は、この内容を実現するため上記①～③に対応する機械語命令を選定する。すなわち、①アドレス1000番地を即値データとして、第１のアドレスレジスタに格納するmov命令、②第１のアド

レスレジスタを用いて*aの内容（ポインタ）をアドレスとして、第２のアドレスレジスタに読み出すmov命令、③第２のアドレスレジスタを用いて、ポインタが指すデータをデータレジスタに読み出すmov命令、の３つの命令を選定する。上記第１、第２のアドレスレジスタは、A0、A1が割り当てられているものとする。これは、シンボルテーブルに記入される。

【００４４】

＜個別命令①の生成＞

さらに、コード生成部７は、これらの複数の命令の中から処理すべき命令（個別命令）を１つ指定し（ステップ２４）、個別命令①に対する命令コードを生成する（ステップ２５）。

このステップ２５における個別命令①の命令コード生成を図３～６のフローを用いて具体的に説明する。

【００４５】

コード生成部７は、ステップ２４で指定された個別命令①がメモリアクセスするロード／ストア命令でなく（図３のステップ３０）、レジスタA0を使用する命令であるのでレジスタ管理部１０にその旨通知する（ステップ３１）。

レジスタ管理部１０は、シンボルテーブルを参照し（図５のステップ５０）、レジスタA0に格納すべき変数がポインタであることから（ステップ５１）、レジスタの全２４ビットが有効となる命令を生成すべきことをコード生成部７に対して指示する（ステップ５３）。

【００４６】

さらに、コード生成部７は、個別命令①が即値データを用いるので即値管理部１１にその旨通知する（ステップ３２）。即値管理部１１は、シンボルテーブルを参照し（図６のステップ６０）、レジスタA0に格納すべき即値がポインタであることから（ステップ６１）、３バイトの即値データを有する命令を生成すべきことをコード生成部７に対して指示する（ステップ６３）。

【００４７】

ステップ５３、６３からの指示に従って、コード生成部７は、個別命令①に対応する、次に示す命令１を生成する（ステップ３３）。

命令１：　　MOV　#001000,A0

この後コード生成部７は、ステップ２３で選定された複数命令のうち、次の命令コードを生成すべき命令②、③が残っているので（ステップ２６）、ステップ２４に進む。

【００４８】

＜個別命令②の生成＞

コード生成部７は、これらの複数の命令の中から処理すべき命令（個別命令）を１つ指定し（ステップ２４）、個別命令②に対する命令コードを生成する（ステップ２５）。

このステップ２５における個別命令①の命令コード生成を図３～６のフローを用いて具体的に説明する。

【００４９】

コード生成部７は、個別命令②がメモリアクセスするロード／ストア命令であるので、メモリ管理部９にその旨通知する（図３のステップ３０）。

メモリ管理部９は、シンボルテーブルを参照し（図４のステップ４０）、第２のレジスタA1に格納すべき変数がポインタであることから（ステップ４１）、メモリアクセスにおけるアクセス幅が３バイトであるロード命令を生成すべきことをコード生成部７に対して指示する（ステップ４３）。

【００５０】

さらに、コード生成部７は、個別命令②がレジスタA0、A1を使用する命令であるのでレジスタ管理部１０にその旨通知する（ステップ３１）。

レジスタ管理部１０は、シンボルテーブルを参照し（図５のステップ５０）、第２のレジスタA1に格納すべき変数がポインタであることから（ステップ５１）、レジスタの全２４ビットが有効となる命令を生成すべきことをコード生成部７に対して指示する（ステップ５３）。

【００５１】

この後、コード生成部７は、個別命令②が即値データを用いないので（ステップ３２）、ステップ３３に進む。

ステップ４３、５３からの指示に従って、コード生成部７は個別命令②に対応

する、次に示す命令２を生成する（ステップ３３）。

命令２：　　MOV　@A0,A1

この後コード生成部７は、ステップ２３で選定された複数命令のうち、次の命令コードを生成すべき命令③が残っているので（ステップ２６）、ステップ２４に進む。

【００５２】

＜個別命令③の生成＞

個別命令③についても上記と同様にして、コード生成部７は次の命令３を生成する（ステップ２４～２６）。この時点で、一時変数t1には、D0レジスタが割り当てられている。

命令３：　　MOV　@A1,D0

以下、上記と同様にして、ステップ２１～２６のループにおいて各中間命令の処理が、ステップ２４、２５（図３、４のフロー）、２６において各個別命令の処理が行われるので、詳細な説明は省略して概要を示す。

【００５３】

<< 中間命令２：t2:=t1+b >>

この中間命令は、「④変数bが割り当てられている1004番地から２バイト読み出して、⑤その内容と一時変数t1とを加算して一時変数t2に格納する。」ことを内容とする。

まず、コード生成部７は、個別命令④に対応してA0レジスタの内容が示す1000番地から4番地離れた番地をロードする命令４を生成する。その際、メモリ管理部９は、シンボルテーブルを参照して（ステップ４０）、個別命令４が整数型データ変数をロードすることから（ステップ４１）、アクセス幅が２バイトのロード命令を生成すべきことを指示する（ステップ４２）。レジスタ管理部１０は、シンボルテーブルを参照して（ステップ５０）、レジスタに整数型データを格納するので（ステップ５１）、レジスタの下位１６ビットが有効となる命令を生成すべきことを指示する（ステップ５２）。この時点で、レジスタD1には変数bが格納されている。

【００５４】

命令４：　　MOV　@(04,A0),D1

次に、コード生成部７は、個別命令⑤に対応してD1レジスタに格納された変数bと、D0レジスタの内容が示す一時変数t1とを加算して、その結果をD1レジスタに格納する命令５を生成する。その際、レジスタ管理部１０は、シンボルテーブルを参照して（ステップ５０）、レジスタに整数型データを格納するので（ステップ５１）、レジスタの下位１６ビットが有効となる命令を生成すべきことを指示する（ステップ５２）。この時点で、一時変数t2にはレジスタD1が割り当てられている。

【００５５】

命令５：　　ADD　D0,D1

<<中間命令３：t3:=t2+1 >>

この中間命令は、「⑥一時変数t2に1を加算して一時変数t3に格納する」ことを内容とする。コード生成部７は、レジスタ管理部１０、即値管理部１１の指示に従って、２バイトの即値#0001とレジスタD1とを加算した結果をD1レジスタに格納する次の命令６を生成する。この時点で、一時変数t3にはレジスタD1が割り当てられている。

【００５６】

命令６：　　ADD　#0001,D1

<<中間命令４：c:=t3 >>

この中間命令は「⑦変数cが割り当てられている1006番地から２バイトの領域に一時変数t3を書き込む」ことを内容とする。コード生成部７は、メモリ管理部９の指示に従って、レジスタA1の内容が示す番地から６番地離れた番地にレジスタD1の内容をストアする次の命令７を生成する。

【００５７】

命令７：　　MOV　D1,@(06,A0)

上記のようにして各中間命令の処理が終了する。その結果、コード生成部７から次のようなオブジェクトコードがメモリ１３に対して出力される。メモリ１３上で、命令１から命令７はそれぞれ、100000番地、100005番地、100007番地、100008番地、10000a番地、10000b番地、10000f番地に配置されるものとする。

【００５８】

命令１：　100000番地　MOV　#001000,A0

命令２：　100005番地　MOV　@A0,A1

命令３：　100007番地　MOV　@A1,D0

命令４：　100008番地　MOV　@(04,A0),D1

命令５：　10000a番地　ADD　D0,D1

命令６：　10000b番地　ADD　#0001,D1

命令７：　10000f番地　MOV　D1,@(06,A0)

これらの命令は、オブジェクトコードを便宜上ニモニック（アセンブリ言語）で表現したものであり、メモリ１３に格納される際は２進数で表される。上記の数値はすべて１６進数である。

以下、メモリ１３に配置されたこのオブジェクトコードについて、マイクロコンピュータにより実行される動作を説明する。

【００５９】

（命令１：100000番地　MOV #001000,A0　）

バス制御回路１６および命令制御回路１７は、プログラムカウンタ１９が保持している値100000をアドレスバス１４に出力し、データバス１５を介して命令１をフェッチして解読する。解読結果に従って演算実行回路１８は、命令１のオペランドで指定されている即値001000を命令制御回路１７から受け取り、レジスタファイル２０の中のA0レジスタに格納する。

【００６０】

（命令２：100005番地　MOV @A0,A1　）

バス制御回路１６および命令制御回路１７は、同様にして、命令２をフェッチして解読する。演算実行回路１８は、A0レジスタの内容を読み出し、それをアドレスバス１４に出力してメモリ１３を読み出し、データバス１５を介して読み出された１６ビットのデータをA1レジスタの下位１６ビットに格納する。続いて、演算実行回路１８は、演算器２１によりA0レジスタの値001000に2を加算し、バス制御回路１６により加算結果001002をアドレスバス１４に出力してメモリ１３を読み出し、データバス１５を介して読み出された8ビットのデータをA1レジス

タの上位8ビットに格納する。

【００６１】

（命令３：100007番地　MOV @A1,D0　）

バス制御回路１６および命令制御回路１７は、命令３をフェッチして解読する。演算実行回路１８は、A1レジスタの内容を読み出し、それをアドレスバス１４に出力し、メモリ１３を読み出す。その後演算実行回路１８は、データバス１５を介して読み出された１６ビットの値をD0レジスタの下位１６ビットに格納する。D0レジスタは、ポインタ変数*aが指し示すデータを保持することになる。

【００６２】

（命令４：100008番地　MOV @(04,A0),D1　）

バス制御回路１６および命令制御回路１７は、命令４をフェッチして解読する。演算実行回路１８は、変位値04を命令制御回路１７から受け取り、演算器２１において読み出されたA0レジスタの値001000と加算し、加算結果001004をバス制御回路１６によりアドレスバス１４に出力しメモリ１３の読み出しを行う。その後演算実行回路１８は、データバス１５に読み出された１６ビットの値をレジスタファイル２０の中のD1レジスタの下位１６ビットに格納する。D1レジスタは、変数bを保持することになる。

【００６３】

（命令５：10000a番地　ADD D0,D1　）

バス制御回路１６および命令制御回路１７は、命令５をフェッチして解読する。演算実行回路１８は、演算器２１においてレジスタファイル２０から読み出されたD0レジスタの値とD1レジスタの値とを加算し、加算結果をレジスタファイル２０の中のD1レジスタに格納する。演算器２１はこの加算を２４ビットで行うが、D1レジスタは下位１６ビットが有効である。D1レジスタは、ポインタ変数*aが示すデータと変数bとの加算値を保持することになる。

【００６４】

（命令６：10000b番地　ADD #0001,D1　）

バス制御回路１６および命令制御回路１７は、命令６をフェッチして解読する。演算実行回路１８は、演算器２１においてレジスタファイル２０から読み出さ

れたD1レジスタの値と、命令制御回路１７から受け取った即値0001とを加算し、加算結果をD1レジスタに格納する。D1レジスタは、ポインタ変数*aが示すデータと変数bと即値0001との加算値を保持することになる。

【００６５】

（命令７：10000f番地　MOV D1,@(06,A0)　）

バス制御回路１６および命令制御回路１７は、命令７をフェッチして解読する。演算実行回路１８は、変位置06を命令制御回路１７から受け取り、演算器２１においてレジスタファイル２０から読み出されたA0レジスタの値001000と加算し、加算結果001006をバス制御回路１６によりデータバス１５に出力するとともに、D1レジスタの下位１６ビットの値をデータバス１５に出力し、メモリ１３に対して１６ビットデータの書き込みを行う。メモリ１３の001006番地には、ポインタ変数*aが示すデータと変数bと即値0001との加算値が書き込まれることになる。

【００６６】

このようにしてＣ言語プログラムがコンパイラ２で翻訳されて生成されたオブジェクトコードが、ハードウェア３で実行される。

以上のように本実施例によれば、コンパイラ２が全ての変数と生成するオブジェクトコードとのメモリ上での配置を２４ビットのアドレスで管理し、マイクロコンピュータが２４ビットのレジスタファイル２０および演算器２１でこれらのアドレスを計算し２４ビットのアドレスバス１４でメモリ１３をアクセスするため、本実施例の情報処理装置はセグメントに分割しない全く均一な１６メガバイトのアドレス空間を実現することができ、従ってＣ言語プログラムを記述するプログラマはセグメントの境界のような空間の不均一性を認識する必要がなく、またセグメントレジスタの操作のようなアドレス計算での性能劣化も発生しない。それに伴ってプログラムの開発効率も向上する。

【００６７】

また、データのビット幅が１６ビットで１６メガバイトのアドレス空間を必要とする応用にとって、本実施例の情報処理装置は、マイクロコンピュータが２４ビットのレジスタファイル２０および演算器２１で構成されるため、従来の３２

ビットプロセッサのように過分なハードウェアによってコストおよび消費電力が増大することがない。

【００６８】

また、マイクロコンピュータの機械語命令の基本語長は８ビットであり、上記の命令１のように命令中にアドレスの即値が伴う場合でもその即値は最大でも２４ビットとなるため、コンパイラ２が生成するオブジェクトコードは、機械語命令の基本語長を１６または３２ビットとしアドレスの即値が最大３２ビットとなる３２ビットプロセッサの情報処理装置に比べて極めて小さくなる。また６４キロバイトのアドレス空間しか扱えない１６ビットプロセッサに比べても、コードサイズの増加の要因が命令に伴うアドレスの即値による最大１バイトだけであるため、オブジェクトコードのサイズはほとんど大きくならない。

【００６９】

本実施例のマイクロコンピュータのデータバス１５の幅は１６ビットであるが、これを２４ビットとした場合と比べると、レジスタファイル２０に格納された２４ビットのアドレスに関する値をメモリ１３に対して読み書きする時のみ実行時間が長くなるだけであり、この性能の低下は、メモリに対して常に３２ビットで読み書きする３２ビットプロセッサの情報処理装置のデータバスのビット幅を１６ビットにした場合に比べて極めて小さい。

【００７０】

なお、本実施例は、マイクロコンピュータのアドレスバス１４、プログラムカウンタ１９、レジスタファイル２０および演算器２１のビット幅を２４ビットとし、コンパイラ２のパラメータ設定部８のポインタ変数のビット幅を２４ビットに設定しているが、これらを必要とするアドレス空間の広さに対応して１７ビットから３１ビットの任意のビット幅にしてもよい。対応は次のようになる。

【００７１】

必要なアドレス空間	ビット幅
１２８キロバイト	１７ビット
２５６キロバイト	１８ビット
５１２キロバイト	１９ビット

１メガバイト	２０ビット
２メガバイト	２１ビット
４メガバイト	２２ビット
８メガバイト	２３ビット
１６メガバイト	２４ビット（本実施例）
３２メガバイト	２５ビット
６４メガバイト	２６ビット
１２８メガバイト	２７ビット
２５６メガバイト	２８ビット
５１２メガバイト	２９ビット
１ギガバイト	３０ビット
２ギガバイト	３１ビット

このようにすることにより、アドレスのビット幅を超す過分なハードウェアがなくなり、応用に応じてコストおよび消費電力の最適化を図ることができる。

【００７２】

また本実施例は、マイクロコンピュータのデータバス１５の幅を１６ビットとしているが、これを２４ビットとしてもよい。

また本実施例は、マイクロコンピュータのアドレスバス１４、プログラムカウンタ１９、レジスタファイル２０および演算器２１のビット幅とコンパイラ２のパラメータ設定部８のポインタ変数のビット幅の設定値とを２４ビットとし、コンパイラ２のパラメータ設定部８のデータ変数のビット幅の設定値を１６ビットとしているが、これらのビット幅を限定するものではない。後者をプログラムが主として扱う型のデータのビット幅（Ｍとする）と等しくし、前者をＭより大きい値（Ｎとする）とすることにより、情報処理装置は、主として扱う型のデータのビット幅で表現できる２のＭ乗バイトの空間を越える２のＮ乗バイトのアドレス空間を扱うことができる。

【００７３】

また本実施例は、パラメータ設定部８でビット幅を設定するデータ変数の型を整数型としているが、これをＣ言語プログラムで主として扱うデータの型に合わ

せていかなる型にしてもよい。

また本実施例は、Ｃ言語プログラムをコンパイルして実行するものであるが、言語を限定するものではない。コンパイラ２をプログラムの記述言語に対応させることにより、情報処理装置はいかなる言語のプログラムでも実行できる。

【００７４】

【発明の効果】

以上説明してきたたように本発明によれば、パラメータ設定手段にデータ幅及びポインタ幅を任意に設定できるので、プログラマにセグメントの境界のような空間の不均一性を認識させることなく、かつセグメントレジスタの操作のようなアドレス計算での性能劣化をも伴うことなく、主として扱う型のデータのビット幅で表現できる空間を超える広さの任意のアドレス空間を扱うことができるという効果がある。

【００７５】

また、組み込み用途のマイクロコンピュータでの応用（アプリケーション）おいては、データのビット幅が１６ビット必要で、かつ、アドレス空間が６４キロバイト以上を必要とするが４ギガバイトもの大きさを必要としないものに対して最適化を図ることができるという効果がある。

また本発明によれば、従来の３２ビットプロセッサの情報処理装置に比べてプログラムのコードサイズが小さく、またデータバスのビット幅が１６ビットの場合でも性能劣化がほとんどない情報処理装置を実現できるという効果がある。

【図面の簡単な説明】

【図１】

本発明の実施例における情報処理装置のブロック図である。

【図２】

同実施例におけるコード生成部７の処理フローを示す。

【図３】

同実施例におけるコード生成部７の処理フローのうち個別処理のフローを示す。

【図４】

同実施例におけるメモリ管理部９の処理フローを示す。

【図５】

同実施例におけるレジスタ管理部１０の処理フローを示す。

【図６】

同実施例における即値管理部１１の処理フローを示す。

【図７】

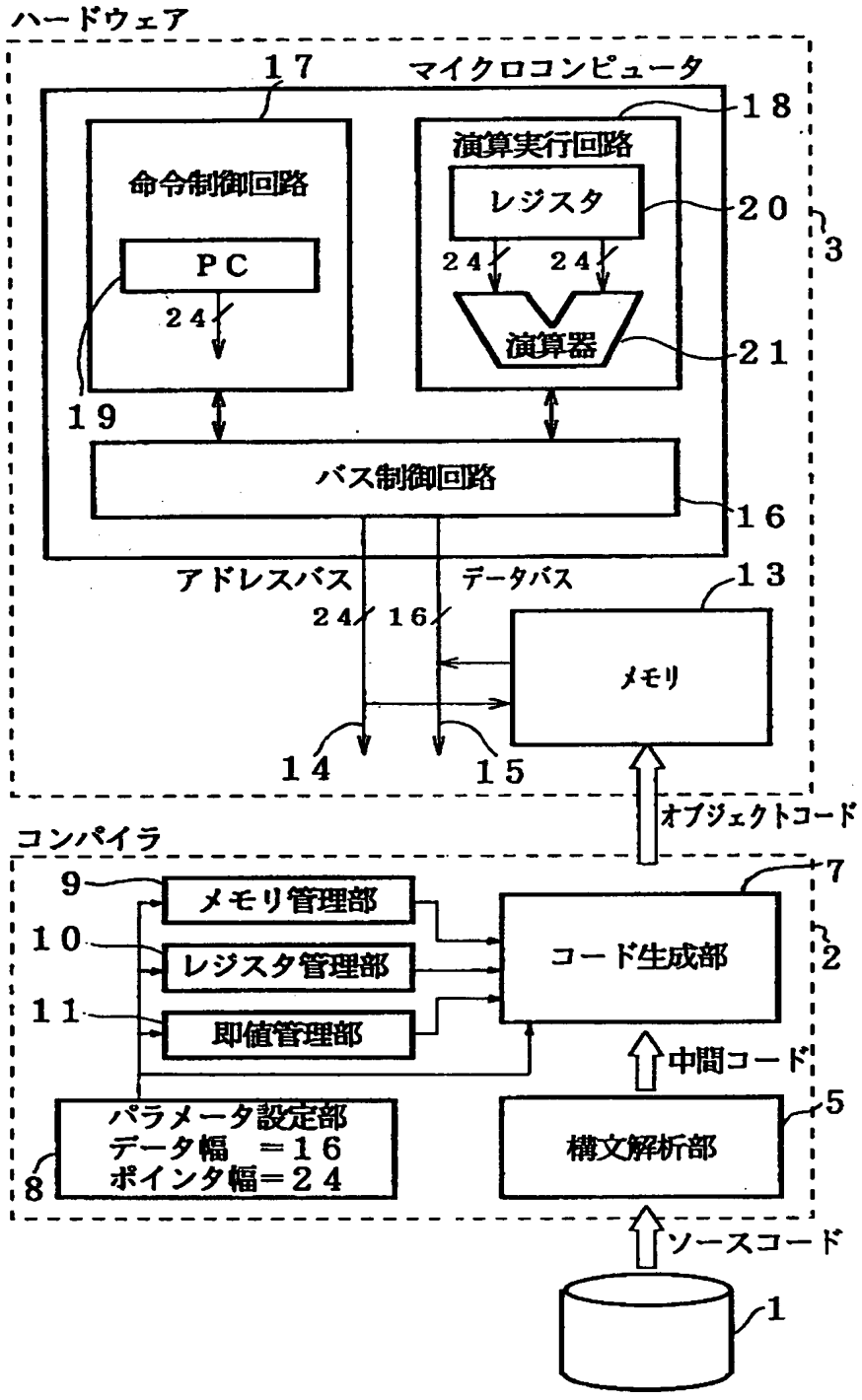
同実施例におけるシンボルテーブルを示す。

【符号の説明】

１　記憶装置
２　コンパイラ
３　ハードウェア
５　構文解析部
７　コード生成部
８　パラメータ設定部
９　メモリ管理部
１０　レジスタ管理部
１１　即値管理部
１３　メモリ
１４　アドレスバス
１５　データバス
１６　バス制御回路
１７　命令制御回路
１８　演算実行回路
１９　プログラムカウンタ
２０　レジスタファイル
２１　演算器

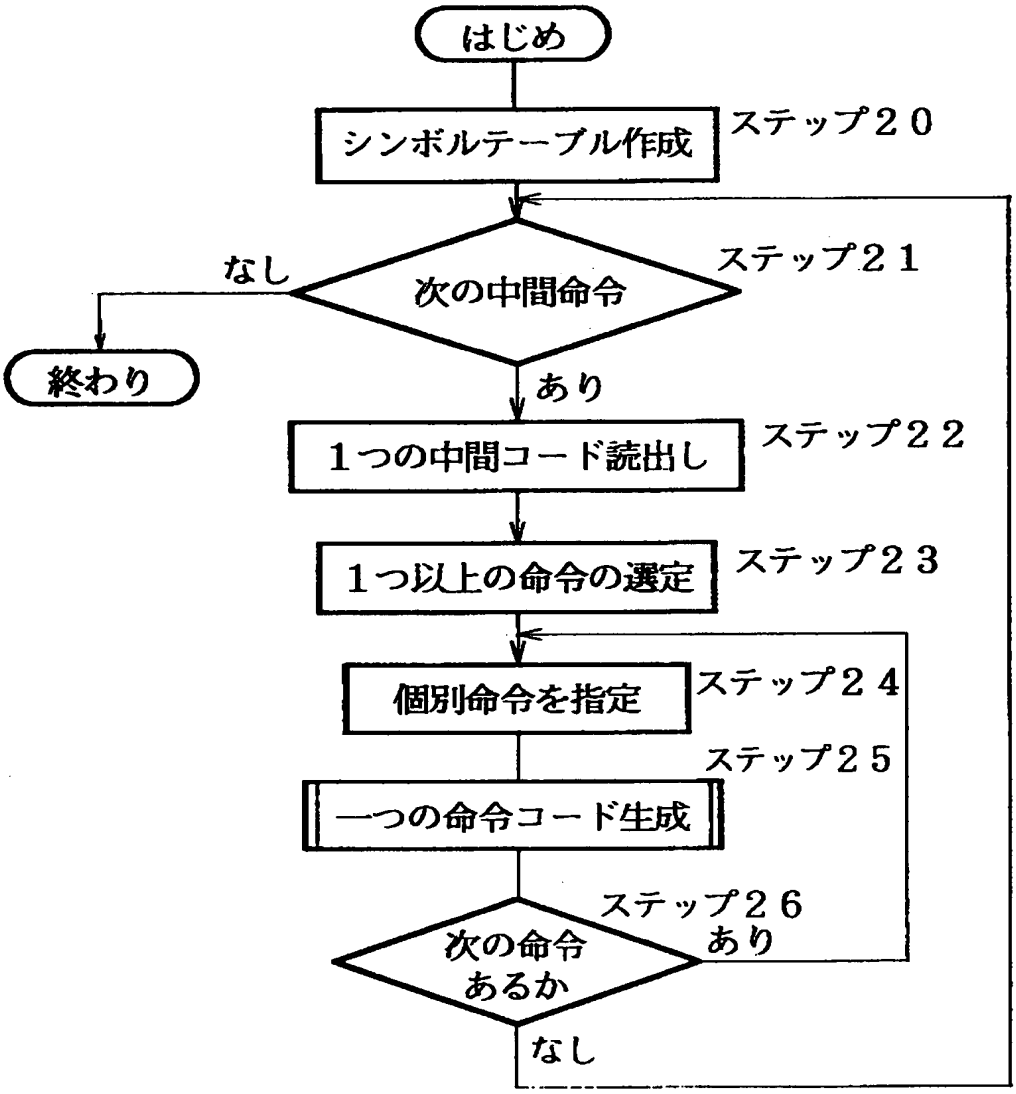
【書類名】　図面

【図１】


ハードウェア
17
マイクロコンピュータ
18
命令制御回路
PC
24
演算実行回路
レジスタ
20
24
24
演算器
21
3
19
バス制御回路
16
アドレスバス
データバス
24
16
13
メモリ
14
15
オブジェクトコード
コンパイラ
9
メモリ管理部
10
レジスタ管理部
11
即値管理部
7
コード生成部
2
中間コード
パラメータ設定部
データ幅　＝１６
ポインタ幅＝２４
8
構文解析部
5
ソースコード
1

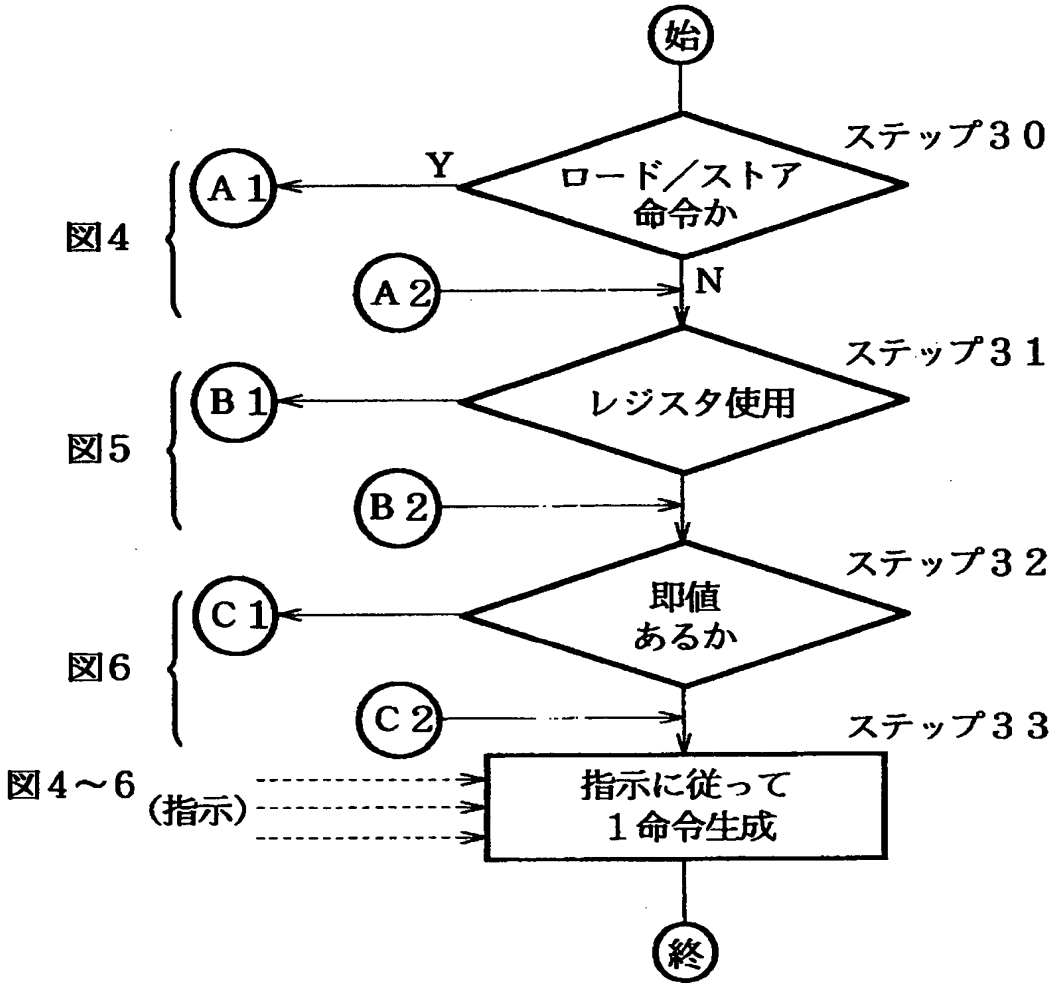
【図２】

コード生成部７の処理フロー

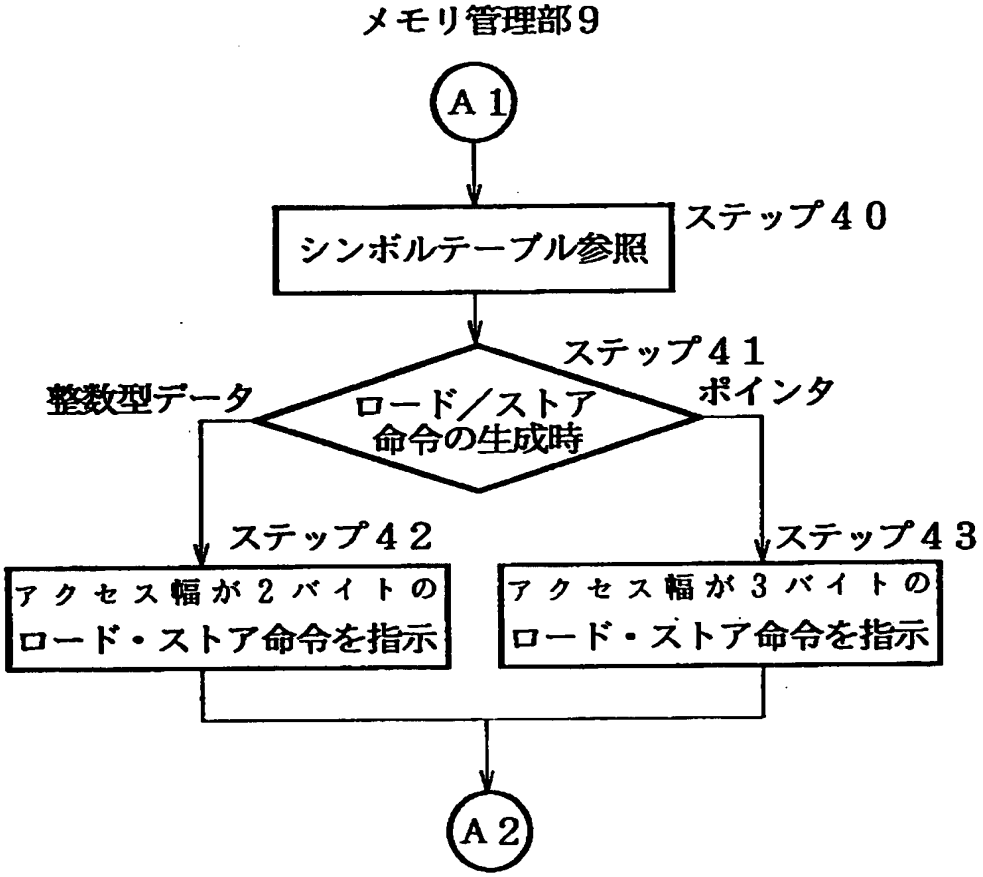


【図３】

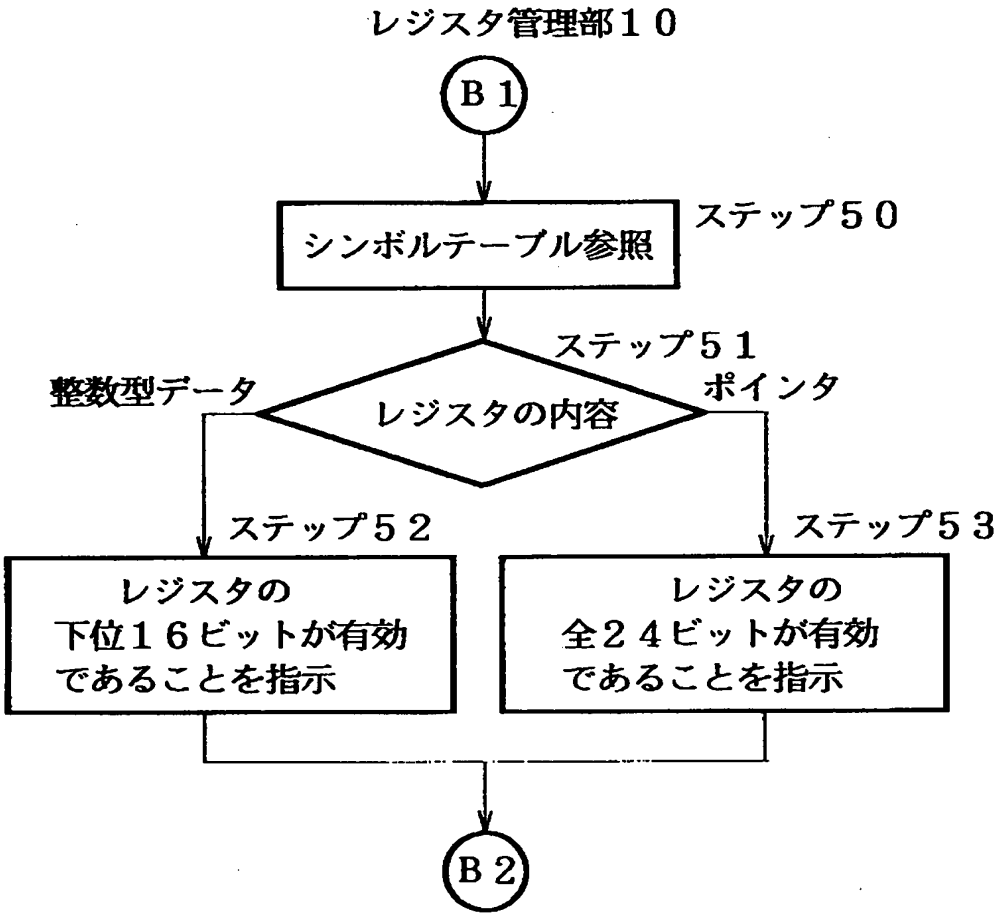
コード生成部のフロー



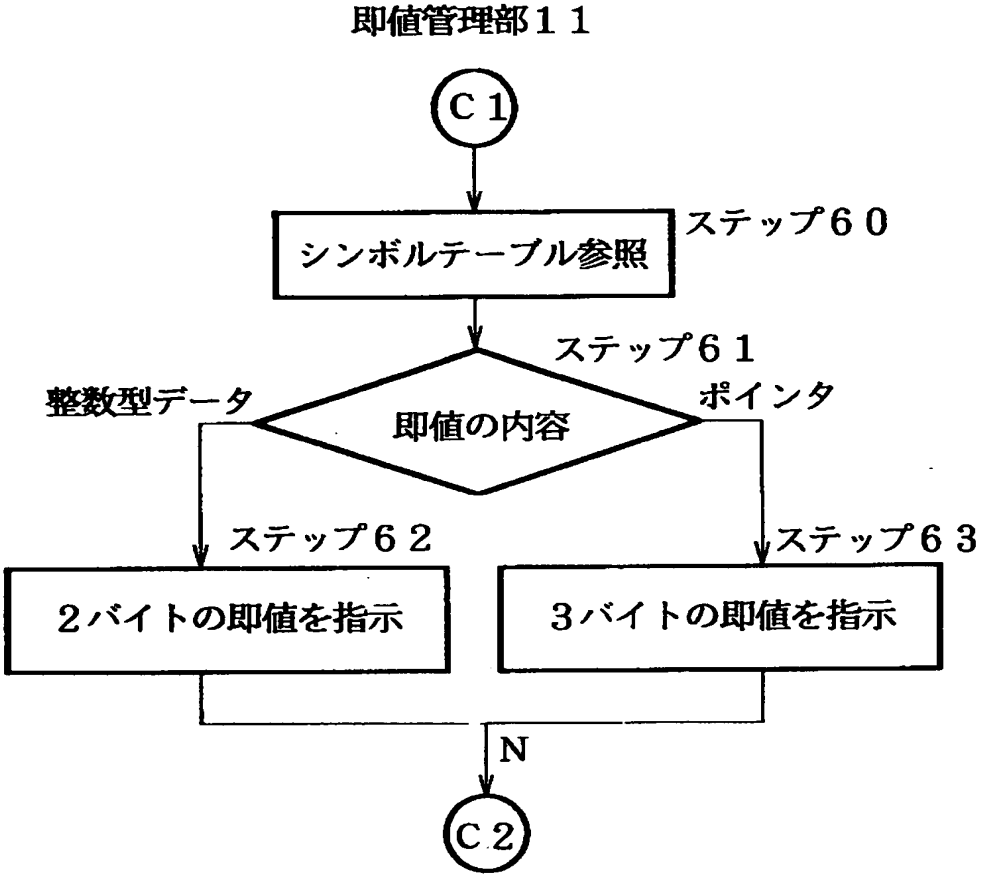
【図４】


メモリ管理部９
A１
シンボルテーブル参照
ステップ４０
ステップ４１
整数型データ
ロード／ストア
命令の生成時
ポインタ
ステップ４２
ステップ４３
アクセス幅が２バイトの
ロード・ストア命令を指示
アクセス幅が３バイトの
ロード・ストア命令を指示
A２

【図５】


レジスタ管理部１０
Ｂ１
シンボルテーブル参照
ステップ５０
ステップ５１
整数型データ
レジスタの内容
ポインタ
ステップ５２
ステップ５３
レジスタの
下位１６ビットが有効
であることを指示
レジスタの
全２４ビットが有効
であることを指示
Ｂ２

【図６】


即値管理部１１
C 1
シンボルテーブル参照
ステップ６０
ステップ６１
整数型データ
即値の内容
ポインタ
ステップ６２
ステップ６３
２バイトの即値を指示
３バイトの即値を指示
N
C 2

【図７】

シンボルテーブル

シンボル	型	バイト数	先頭アドレス	レジスタ
*a	ポインタ	3	１０００	
b	整数	2	１００４	
c	整数	2	１００６	
t１	整数	2		
t２	整数	2		
t３	整数	2		

【書類名】　要約書

【要約】

【目的】　本発明は、データのビット幅が１６ビットで１６メガバイト程度のアドレス空間を必要とする応用において、好適したコンパイラおよびマイクロコンピュータを提供することを目的とする。

【構成】　プログラム変換装置において、メモリ管理部９、レジスタ管理部１０、即値管理部１１からの指示により、コード生成部７は、生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成する。アドレスを表す変数である場合、パラメータ設定部８が保持するアドレスのビット幅を有効とする命令を生成する。マイクロコンピュータは、Ｎ（24）ビット長のプログラムカウンタにより２のＮ乗（16Mバイト）のアドレス空間を利用でき、実行手段によりＮ（24）ビット長の演算、及びデータのビット幅であるＭ（16）ビット長の演算を実行し得る。

【選択図】　図１

【書類名】　職権訂正データ
【訂正書類】　特許願

＜認定情報・付加情報＞
【特許出願人】
【識別番号】　000005821
【住所又は居所】　大阪府門真市大字門真１００６番地
【氏名又は名称】　松下電器産業株式会社
【代理人】　申請人
【識別番号】　100090446
【住所又は居所】　大阪府大阪市北区豊崎３丁目２番１号　淀川５番館５Ｆ　中島国際特許事務所
【氏名又は名称】　中島　司朗

出　願　人　履　歴　情　報

識別番号　　〔０００００５８２１〕

１．変更年月日　　１９９０年　８月２８日
［変更理由］　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社